

### REMARKS

Claims 1, 3 to 14, and 16 to 28 are pending in this application. Of these, claims 1 and 14 are independent.<sup>1</sup> Favorable reconsideration and further examination are respectfully requested.

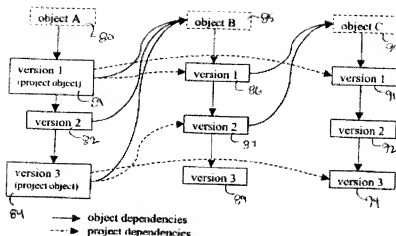
All of the claims were rejected over (i) U.S. Patent Publication No. 2004/0126750 (Theilmann) in view of U.S. Patent Publication No. 2004/0002049 (Beavers), and (ii) U.S. Patent Publication No. 2002/0168621 (Cook) in view of U.S. Patent Publication No. 2006/0059253 (Goodman).

Independent claim 1 has been amended to recite a method, performed by one or more processing devices, for use in an electronic learning system that stores information as learning objects. The method comprises designating a target learning object as a project object, and storing version dependency data in the project object. The project object is an object that is separate from the first object and the second object. The version dependency data identifies at least a version of a first object upon which the project object directly depends, and a version of a second object upon which the project object indirectly depends. The first object stores dependency data identifying the second object upon which the first object depends. The first object does not store version dependency data identifying the version of the second object upon which the first object depends. The version of the second object upon which the first object depends is determined by the version dependency data in the project object.

An embodiment of claim 1 is shown in Fig. 8 of the application, which is reproduced below.

---

<sup>1</sup> The Examiner is urged to independently confirm this recitation of the pending claims.



As can be seen from Fig. 8, a project object 81 stores version dependency data, which is identified by the dashed lines. The version dependency data identifies the versions of other objects, here object 85 and object 90, upon which the project object 81 depends. In this example, object 85 exists in three versions: 86, 87 and 89; and object 90 exists in three versions: 91, 92 94. The version dependency data identifies the version of the object upon which the project object depends. A first object, e.g., object 85, contains dependency data, which is identified by solid lines. The dependency data identifies a second object, e.g., object 90, upon which the first object 85 depends. The first object 85, however, does not include version dependency data. That is stored in the project object. Therefore, when determining an object upon which the first object 85 depends, one looks to the dependency data in the first object 85 to identify second object 90, and infers the version of object 90 from the version dependency data in project object 81. That is, since project object 81 depends on version 1 91 of second object 90, one can infer

that first object 85, upon which project object 81 depends, also depends on version 1 91 of second object 90. Thus, version dependency data is not needed in first object 85.

There is nothing akin to the foregoing arrangement that is disclosed in the cited art.

As explained in prior amendments, Cook fails to disclose or to suggest the features of claim 1, in particular, storing version dependency data in the project object, where the version dependency data identifies at least a version of a first object upon which the project object directly depends, and a version of a second object upon which the project object indirectly depends. In fact, it was admitted on page 14 of the Office Action that Cook does not disclose version dependency data. Therefore, Cook could not possibly disclose or suggest the foregoing features. Nevertheless, the Office Action cites paragraphs 0078 of Cook for allegedly disclosing "version dependency". In this regard, the cited paragraphs of Cook do not even disclose dependencies between objects (in fact, the word "depend" is not even used in these paragraphs). Applicants therefore remain at a loss as to how those portions of Cook could possibly disclose or suggest the foregoing features of claim 1.

Goodman was cited for its alleged disclosure of object and data dependencies in its paragraphs 0138, 0170, 0173, 0316 and 0378.

In this regard, Goodman describes a system that keeps track of different versions of software, e.g., through version control services, as described below.

As with standard development code, when media content is created and edited, the version control services maintain a revision history of changes. This way, if it is necessary to revert to an original piece of media content, it is not necessary to go all the way back to the original source (which in the case of finding an image in a CD-ROM library containing 10,000 images, for example, could be a difficult task). In practice, this may mean storing the original and final copies

of media (especially where volume is an issue). For this reason, a process for managing multiple versions of media content is put into place in the preferred development architecture 500.<sup>2</sup>

The Office Action cites paragraphs 0316 and 0378 of Goodman for allegedly describing storage of version dependency data in a project object and dependency data, but not version dependency data, in objects that depend from the project object. These paragraphs are reproduced below for discussion purposes.

[0316] As illustrated in FIG. 26, the information management tools 602 is also responsible for object management 646. Object management tools provide capabilities for viewing objects, their methods and attributes, and the dependencies between these objects. Object management tools also provide specific analysis tools, in order to understand interdependencies between the core classes and the components. When classes and components are modified, impact analysis tools are required to see where the modified entity is being used, allowing them to understand what is the overall impact of the change. This is more complex than with traditional systems as a veritable spider's web of dependencies between classes, components, and applications may ensue. In addition, OO features such as inheritance and polymorphism make tracking down dependencies with simple text search tools much more difficult.

[0378] Process modeling tools 600 provide a graphical depiction of the business functions and processes being supported by a system. The tool(s) selected must support the modeling techniques being used in the development methodology; these include process decomposition, data flow and process dependency. Process modeling is a method for clarifying and communicating the business design of the system. The process model can provide an effective means of bringing people together, creating a shared vision of how the business is to function. A process model provides a means of standardizing a set of similar processes, which exist, for example, at different branches of the business.

As explained previously, paragraph 0316 does describe dependencies among objects, and "specific analysis tools...to understand interdependencies between the core classes and the components". Paragraph 0378, on the other hand, merely mentions "process dependency". Significantly, however, neither these paragraphs, nor the remainder of Goodman, describes exactly how the dependencies are analyzed or managed in connection with different versions.

---

<sup>2</sup> Paragraph 0176

In particular, Goodman does not disclose or suggest that the project object is an object that is separate from the first object and the second object, and that the version of the second object upon which the first object depends is determined by the version dependency data in the project object, which is not stored in the first object.

Paragraphs 0138, 0170 and 0173 of Goodman were also cited for their alleged disclosure of version control and storing dependency data. These paragraphs are reproduced below.

[0138] The environment management process services 560 supports the environment where management processes are performed, and where systems are being built. The release management process services 562 manage the simultaneous development of multiple releases. The configuration management process services 564, often closely linked with release management, cover the version control, migration control, and change control of system components, such as code and its associated documentation. The problem management process services pertain to the problem tracking and solution process.

[0170] Another important distinction maintained by the folder management services 572 is the one between work in progress and completed documents that have been approved. This distinction is supported by a folder structure with carefully chosen access rights. An example of how the folder management services 572 maintain documentation for an application is illustrated in FIG. 21. As illustrated, documentation is divided into two general categories, work-in-progress and completed documentation. Work-in-progress can further be divided into teams, wherein each team maintains its own technical documentation and user documentation. Completed documentation is broken down by versions; including a current version folder and a previous version folder that contain technical documentation and user documentation.

[0173] The three major processes that are required to support media content management are: storage management services; metadata management services; and version control services. Storage management services deal with the methods of storing and retrieving media content. The cost of data storage may be decreasing, but it is still the case that for large volumes of media it is often uneconomical to store everything on-line. For this reason, processes are implemented with the storage management services that manage where data is stored, and how it is transitioned from one location to another within the netcentric computing system 10. There are three ways data is stored in the present invention: on-line (Instant access, for example, hard disk); near-line (delayed access, for example, CD-ROM jukebox); and, off-line (manual access, for example, CDs or tapes on shelves).

The foregoing paragraphs were cited without any explanation. Paragraph 0138 does mention version control, which is admittedly disclosed in Goodman. However, the *claimed method* of

performing version control is not disclosed. Applicants respectfully remind the Examiner that claim 1 is claiming more than simply version control; it is claiming a specific type of version control, which is clearly not shown in the art. Paragraphs 0170 and 0173 likewise allude to version control but, again, not the type that is claimed.

For at least the foregoing reasons, Applicants submit that even if Goodman were combined with Cook in the manner suggested, the resulting hypothetical combination would fail to disclose or to suggest the features of claim 1.

Turning to the rejection over Theilmann and Beavers, page 8 of the Office Action admits that Theilmann does not disclose "storing of the versions and their dependency explicitly as claimed". Beavers was cited to make up for this deficiency of Theilmann. In particular, the Office Action cites paragraphs 0149 and 0174 through 0177 of Beavers.

Paragraph 0149 merely describes that multiple versions of the presentation data are sent in case some packets are lost over the network. Version, in this context, means multiple copies of the same presentation, not different versions of objects. Paragraphs 0174 through 0177 are reproduced below.

[0174] The last menu item is the help item 534. Selecting the help menu item 534 causes a drop-down list (not shown) to appear in the slideshow window. In tested versions of the present system, this list included only one listing--namely an "About" link. Selecting the about link causes an information box to appear in the slideshow window. This information box is a standard "about" box and includes information such as the title of the program and its version number, copyright data, the licensee's name, and perhaps links to other informational windows. While the tested embodiments were limited to just the about link, this need not be the case. Links to various help directories, as well as links to "on-line" support sites, could be included.

[0176] Referring again to FIG. 7, the functions of the aforementioned control bar 520 will now be described. In general, the control bar is made up of a series of control buttons and a selection field that are used by the presenter to run the presentation slide show (and so inactive if the viewer role has been selected under the options menu item). To this end there is a previous slide button 536 and a next slide button 538. When selected, these buttons 536, 538 respectively

change the currently displayed presentation slide to the previous or next slide in the slide deck. In addition, there is a slide selection field 540 that lists the name of the currently selected slide, and which has a drop-down arrow 542 at one end. When the arrow 542 is selected a drop-down list (not shown) appears in the slideshow window listing all the slides in the presentation by name and in the order from top to bottom which they are scheduled to be shown. When a name is selected from the list, the presentation slide associated with that name is displayed in the slide sector of the slideshow window. In addition, if there are too many slides to list in the space allocated to the drop-down list, a standard graphic scroll bar is included along one side of the list so that the user can scroll up or down the list to display the hidden slides. It is note that the names of the slides can be related to a title on the slide, or could be defaulted to list the slide by its sequence number in the slide deck (e.g., "Slide 3").

[0177] The control bar 520 also has three scribble related buttons. The first of these buttons 544 (which is labeled "WB" as short for whiteboard in tested versions of the slideshow window) activates the previously described whiteboard function where only the scribble is displayed in the slide display sector 502. Thus, the whiteboard button 544 represents a shortcut to using the previously described menu options. When the whiteboard button 544 is selected, its label changes. In tested versions of the slideshow window the label changed to an "S" (short for Slides). When the presentation is in the whiteboard mode, selecting the whiteboard button 544 (then labeled with an "S") changes the presentation back to its slide-with-scribble display mode. It is noted that this latter mode is the default mode. There is also a clear ink button 546, which is label with a "C" in tested versions of the slideshow window. The clear ink button 546 clears all the annotations entered by the presenter with the aforementioned tablet PC in association with the currently displayed slide, or if a space which was initially blank is currently being displayed, the button 546 clears any annotations added in that space by the presenter. It should be noted that this button 546 is also a short cut to the "Clear Current Slide" command found under the file menu item 524. Finally, there is a slide minimizing/maximizing button 548 in the control bar 520. In tested versions of the slideshow window, the default mode for the presentation is for the entire slide display sector 502 to be filled with the image of the current slide (i.e., corresponding to the 100 percent setting in the slide size command of the window menu item 532). As such the minimizing/maximizing button 548 is initially configured to minimize the slide size. In tested versions of the slideshow window, the button 548 is labeled "Min" at this the initial stage, and selecting it minimized the slide size to 50 percent of its initial width and height. In addition, the minimized slide is displayed in the upper left region of the slide display sector. Further, when the minimizing/maximizing button 548 is selected with the slide size maximized, the button label changes to "Max". In this configuration, the slide size is once again maximized when the presenter selects the button 548. Here again the slide minimizing/maximizing button 548 acts as a short cut for the slide size and slide centering commands under the window menu item 532.

These paragraphs do not describe version dependency at all. Paragraph 0174 indicates that a version number of a program is displayed. Paragraphs 0176 and 0177 merely indicate what occurs in "tested versions" of its slideshow window.

For at least the foregoing reasons, Applicants submit that even if Beavers were combined with Theilmann in the manner suggested, the resulting hypothetical combination would fail to disclose or to suggest the features of claim 1.

Amended independent claim 14 is a computer program product claim that roughly corresponds to independent claim 1, and is also believed to be patentable.

The dependent claims are also believed to define patentable features of the invention. Each dependent claim partakes of the novelty of its corresponding independent claim and, as such, each has not been discussed specifically herein.

It is believed that all of the pending claims have been addressed. However, the absence of a reply to a specific rejection, issue or comment does not signify agreement with or concession of that rejection, issue or comment. In addition, because the arguments made above may not be exhaustive, there may be reasons for patentability of any or all pending claims (or other claims) that have not been expressed. Finally, nothing in this paper should be construed as an intent to concede any issue with regard to any claim, except as specifically stated in this paper, and the amendment of any claim does not necessarily signify concession of unpatentability of the claim prior to its amendment.

In view of the foregoing amendments and remarks, Applicants respectfully submit that the application is in condition for allowance, and such action is respectfully requested at the Examiner's earliest convenience.

Applicants' undersigned attorney can be reached at the address shown below. All telephone calls should be directed to the undersigned at 617-521-7896.



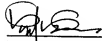
Applicants : Wolfgang Theilmann, et al.  
Serial No. : 10/809,873  
Filed : March 25, 2004  
Page : 19 of 19

Attorney's Docket No.: 13909-161001  
Client Ref.: 2004P00116US

Please apply any fees or credits due in this case to Deposit Account 06-1050 referencing  
Attorney Docket No. 13909-161001.

Respectfully submitted,

Date: November 8, 2007



Paul A. Pysher  
Reg. No. 40,780

Fish & Richardson P.C.  
225 Franklin Street  
Boston, MA 02110-2804  
Telephone: (617) 542-5070  
Facsimile: (617) 542-8906

21779733.doc